Generating Realistic *Impressions* for File-System Benchmarking

Assessing Reproducibility and Fidelity

Puneet Mehrotra
Assignment 1 - CPSC 508 (Winter 2019)
puneet89@cs.ubc.ca

1 Critique

This paper was published in FAST 2009, where it won the best paper award. Impressions was a project at the Advanced Systems Lab at the University of Wisconsin - Madison that's headed by the Arpaci-Dusseaus. This group holds a formidable reputation for carrying interesting and novel research in file and storage systems - a reputation that I firmly believe in [1]

Impressions was a project that was motivated by the lack of formal benchmarking practices in the space of file systems research. This is a well-identified problem, and there has been much research that has tried to address, or at the least, quantify this problem. The problem has been targeted from two different, yet synergistic directions:

- The creation of more *representative workloads* for benchmarking experiments. This involves the study of access patterns on production filesystems and creating workload generators that mimic these access patterns. This appears to be a crowded research space, as can be seen by the sheer number of benchmarks studied by [3].

- Creating more representative filesystem state on which to run benchmarks. This is a combination of the memory state, the on-disk layout/ fragmentation of the filesystem, and the metadata of the filesystem image itself. The cache effects and fragmentation effects have been well studied in prior research. The file-system images used for these studies have been based on ad-hoc and inaccurate assumptions. This is the central issue that this work seeks to address

The authors point out filesystem researchers rely upon non-standard and often arbitrary filesystem images to test the performance of their system, despite there being an abundance of empirical studies of file system contents and metadata. It is, therefore, difficult to verify the performance of the filesystem in a way that is demonstrably independent of the effects that can arise from the actual file-system image used for evaluation.

Impressions is a framework to generate *representative, and statistically accurate file system images while ensuring complete reproducibility of the image*. The framework uses a lot of statistical methods to create the image, and the parameters used to create this image can be tweaked. Also, given the key parameters, it becomes trivial to reproduce the test image, thereby ensuring easy and robust reproducibility.

1.1 Experimental Setup

There is no experimental setup that is explicitly mentioned in the paper. This is very surprising to me since they mention the time it takes to complete desktop search on the generated image. Without any information about the machine and the software on which this experiment was run, it becomes a little difficult to believe and reproduce their results. There is only one explicit system information that is mentioned which is the filesystem that was used to create the generated system on. They mention that they rely on *debugfs* for identifying on-disk layout which is used to modulate fragmentation of the generated filesystem (ext2 or ext3). Even this fact is hidden deep in the section on Disk Layout and Fragmentation, and not explicitly stated as should have been the case.

1.2 Research Execution

The research methodology in this paper is quite sketchy. I think that the methodology design was very elegant, but at the same time, it was presented in a very unsatisfactory manner. I have the following issues with their research execution:

1. In Figure 1, they mention the difference in relative time taken for `find /` on images with different on-disk layouts (fragmentation). The only details

[1] at the time of writing this section.

about this experiment are mentioned in the figure label. This is the only experiment where they actually demonstrate the effect of fragmentation on the filesystem image, and yet they do not explicitly mention the filesystem(ext2 or ext3), or any other system information that is needed to reproduce this result.

2. Table 2 presents the *default* values that the impressions framework assumes. While this is great from a theoretical point-of-view, this does not map well to the input file that the framework uses to accept these parameters. There is scant documentation provided to explain those parameters. While this does not impact the validity of the results - the inputfile shipped with the tarball has all the params already tuned to generate the filesystem image used for the experiments in figure 2.
3. They determine the goodness of their simulation by comparing how closely they can recreate the metrics from the dataset in [2]. This is a good metric for this work. However, it does not reliably convey the effort involved in finding the parameters to make the generative model fit so perfectly with the desired model.
4. The dataset in [2] is based on traces from *Windows PC* machines that were, as the authors pointed, drawn from a homogenous sample. This is not generalizable to other filesystems or even to other workloads (example: a webserver workload). This makes me feel like this paper reports the authors' experience in making the generative model match an arbitrary dataset, and this eclipses the claim of representativeness.

2 Metacomments

2.1 Scope for Improvements

The points noted in subsection 1.2 erode the believability of the results. I thought there is a great sense of evasiveness in this paper. The lack of details about the experimental setup and shallow description of experiment designs very troubling. The paper focuses way too much, and understandably so on the tweaking of the parameters to make the curves fit. Given the overall lack of details in the paper, it appears to have been done with the intention of adding more hay to the haystack to prevent one from finding the needle. There is a lot of scope for improvement here.

- There must be more commentary on the actual experiments being performed in this paper. I can understand that there might be space constraints, but this can be a part of an appendix or an online manual. Without this critical information, it becomes impossible to follow and recreate.

- Information is critical for any kind of systems paper, especially a measurement paper. It is an unpardonable offense to not report these numbers.

- There is something missing in the narrative of this paper. The motivation is solid, and the generative model is elegant. However, somewhere along the line, this paper became one about statistics and how closely they could model a dataset. I would even have accepted that if this was an experience paper which detailed the travails of the research group as they tried to recreate the create the generative model. This paper reads neither like a design paper nor like an experience paper and loses the sense of urgency and purpose midway through. This paper could have done with better editorship.

2.2 Lessons Learned

The lessons I take from this paper are fairly straightforward. When creating and evaluating a system, the actual details about the system and the experiment design used to study it are indispensable. These details not only help the reader reason about the work, but also bolsters the confidence in the findings presented. Computer Science researchers are a sceptic bunch, and there must be enough relevant details provided in a paper to not arouse their suspicion or, quite possibly, wrath.

3 Reproducing Results

For this assignment, I tried to recreate the experiments described in Figures 1 and 2. I now describe my attempts to actually understand and recreate these experiments. Seeing how this is a filesystem paper, and reasoned that there should be no kernel version dependency involved. I ran all experiments on a Ubuntu 16.04 (running a 4.15.0-43 kernel) on an x86-64 machine with 16GB RAM and a 1TB SSD. The disk has been partitioned with an ext4 filesystem.

- Impressions is available for download on the ADSL website [1] as a tarball. On trying to compile the code, I ran into many compilation errors. Most seemed to come from the code that was used for Monte-Carlo Simulation, and some inconsistencies

in the C++ standard used. I fixed these and then tried to run the code.

- On trying to run the code, I ran into segmentation faults. On closer inspection, it appeared to be coming from a buffer overflow when trying to create the pathname for the file generation. Several hours of debugging later, I realized that this is happening because of uninitialized arrays being used for the pathname generation while creating the files. I fixed this and then was able to do a non-crashing run of the program.

- Next I tried to create a filesystem to run the experiments mentioned in Figure 1. I was able to demonstrate the effect of caching by following the following steps:

    1. On a newly rebooted machine, try to find a file that I knew is at some levels deep (6 in this instance). Record the time taken for this.
    2. Read the filesystem contents by running `find / ` on the generated image. This caches the filesystem metadata.
    3. Now repeat the search for the same file as in step 1. This should be faster.

- I also tried to recreate the effect of varying fragmentation in the on-disk layout, but I could not create a flat tree or a deep tree layout. I later learned that this was because they can only support this on an ext3/ext2 filesystem, and I was using ext4.

I spent far too much time in trying to get the fragmentation layout working, but I did not succeed. *Lesson learned: read the fine print. Also, do the bigger and more important experiments first.*

Next, I tried to reproduce the results in Figure 2. To do this, I used the stock inputfile they provided with the tarball and created the filesystem image. All experiments were done using the `find` utility. Again, there was not enough clarity in what the authors meant by Files by size(Fig 2b) versus Files by Containing Bytes(Fig 2d). In the absence of any kind of information, [2], I assumed the following:

1. File by Size means the distribution of the files in the filesystem as measured in block size on disk. To find this I planned to use the `du` utility.
2. File by Containing Bytes refers to the actual number of bytes of content in a file. To measure this I planned to use the `ls -lh` utility.
3. I did not get as far as finding the file distribution by extension types, but I planned to treat `.so` and `.a` files as equivalent to `dlls` and executable binaries (files with execute permissions in ls output) as equivalent to `exe` files.

The experiments for measuring this were quite straightforward: I ran find repeatedly on the root location of the image while controlling the maxdepth parameter and logging the file sizes (on disk and byte size), the number of subdirectories of each directory, and the number of directories at each level.

4 Results

4.1 Impact of Directory Tree Structure

The time taken to find the file in an uncached vs a cached filesystem image is shown in table 1. This constitues a speedup of 22.7X which is similar to what is reported in Figure 1 of the impressions paper.

Table 1: Results of Caching on the time to find a file

	Uncached	Cached
Real Time to find file at depth 6	0m0.638s	0m0.028s

4.2 Accuracy of Impressions in recreating file system properties

The graphs from the experiments as mentioned in Figure two are shown in Figures:1, 2,3

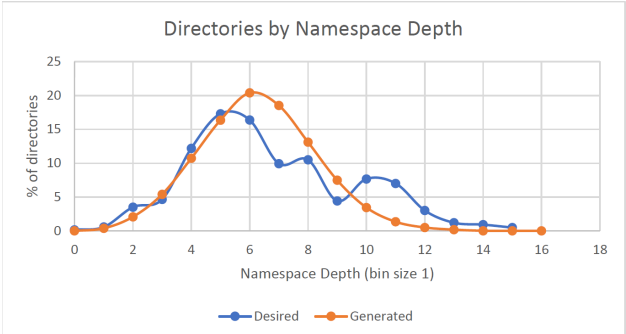

Figure 1: Directories by Namespace Depth

[2] It might be common knowledge in the filesystem research community, but it seemed opaque to me.

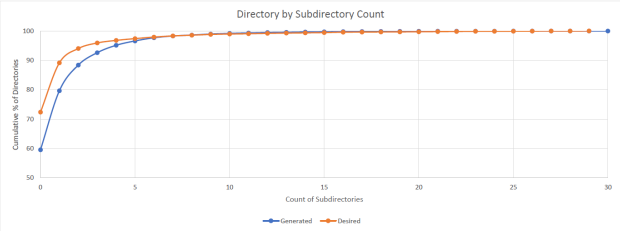

Figure 2: Directories by Subdirectory Count

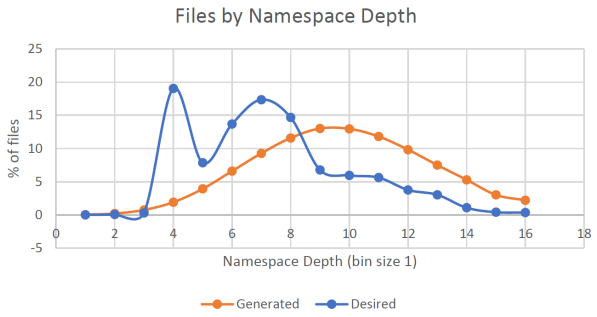

Figure 3: Files by Namespace Depth

These results are not quite as close to the impressions results as I had expected them to be. I have a strong intuition that this is because I am comparing these results to a linux filesystem, and am including the *many* executable files I have littered around in my home directory - I have many kernel source trees in the filesystem and that might explain some of the wierdness.

References

[1] The advanced systems laboratory (adsl) software. `http://research.cs.wisc.edu/adsl/Software/Impressions/`, 2009.

[2] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *ACM Transactions on Storage (TOS)*, 3(3):9, 2007.

[3] V. Tarasov, S. Bhanage, E. Zadok, and M. Seltzer. Benchmarking file system benchmarking: It* is* rocket science. In *HotOS*, volume 13, pages 1–5, 2011.